

Évaluation de la performance de TCP Prague et de l'architecture L4S

Projet de Recherche Innovation Master

Mardi 30 juin 2026

Étudiants : Emre Ucar, Thomas Sauvage, Timothée Fisher

Encadrants Télécom Paris : Jean-Louis Rougier, Nadia Boukhatem

Encadrants Bouygues : Davor Males, Sébastien Bouchez



1 Introduction

Présentation du projet

PRIM (Projet de Recherche Innovation Master) :
Projet de recherche encadré par des professeurs de
Télécom Paris et mené en parallèle des cours.

Notre projet

Étudier **L4S**, une architecture Internet permettant
une **latence faible** et une **perte de paquets
réduite** en cas de congestion sans entrer en
conflit avec les technologies actuellement
déployées.

Standardisée pour la première fois en 2023, l'architecture L4S est actuellement supportée par de nombreux acteurs :



Ce que L4S promet d'apporter

Application	Mauvaise latence	Perte de paquets
Visio-conférence	Délai important entre les questions et réponses	Image et son saccadé, et de qualité détériorée
Cloud-gaming	Délai entre le mouvement de la souris physique et son effet	Image du jeu saccadé, et de qualité détériorée, jeu injouable
Chirurgie à distance	Mouvements exécutés en retard	Commandes non exécutées

Objectifs de L4S : Réduire les problèmes de latence et de pertes de paquets liés à la congestion, ouvrant de **nouveaux usages**

La promesse est-elle tenue ?

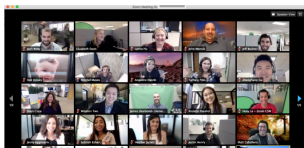


Fig. 1. – Visio-conférence



Fig. 2. – Cloud-gaming



Fig. 3. – Chirurgie à distance

Table des matières

1	Introduction	2
2	Présentation de L4S et TCP Prague	6
3	État de l'art	13
4	Nos contributions	20
5	Nos résultats	28
6	Conclusion	39

2 Présentation de L4S et TCP Prague

Algorithmes de contrôle de congestion TCP

Quel nombre de paquets envoyer simultanément ?

Fenêtre de congestion : Le nombre maximum de paquets envoyables par le client avant de recevoir un ACK / timeout

Différents algorithmes de congestion: **Reno** (RFC 6582 [1]), **Cubic** (RFC 9438 [2]), **BBR** (Draft [3])

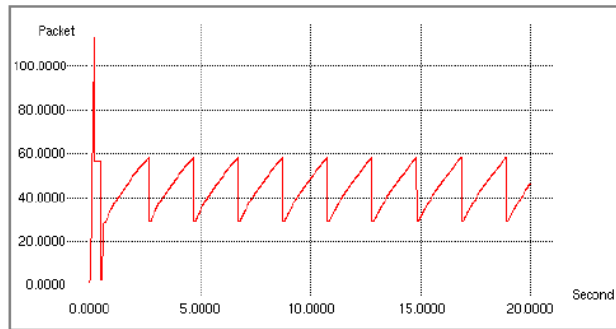


Fig. 4. – Évolution de la fenêtre de congestion typique de l’algorithme TCP Reno

Limites actuelles de TCP

Objectifs

Maximiser le débit, minimiser la latence les pertes

- Augmentation de la taille de la queue : problème de latence dans le passage à l'échelle (**bufferbloat**)
- Maximisation de l'utilisation du lien: problème de *fairness* avec les autres algorithmes (**Reno** comme référence)

Contraignant dans le cas de beaucoup d'applications plus lourdes en besoins Internet (appels vidéo, streaming, cloud-gaming...)

ECN

Explicit Congestion Notification : (RFC 3168, Septembre 2001 [4]) extension TCP/IP permettant de signaler une congestion sans *drop* de paquet, en les marquant à l'aide de bits dédiés dans le header.

Utilisé par exemple dans **DCTCP** (Data-Center TCP, RFC 8257 [5])

Inutilisable sur un réseau public car l'utilisation du lien est trop agressive pour être *Reno-friendly*

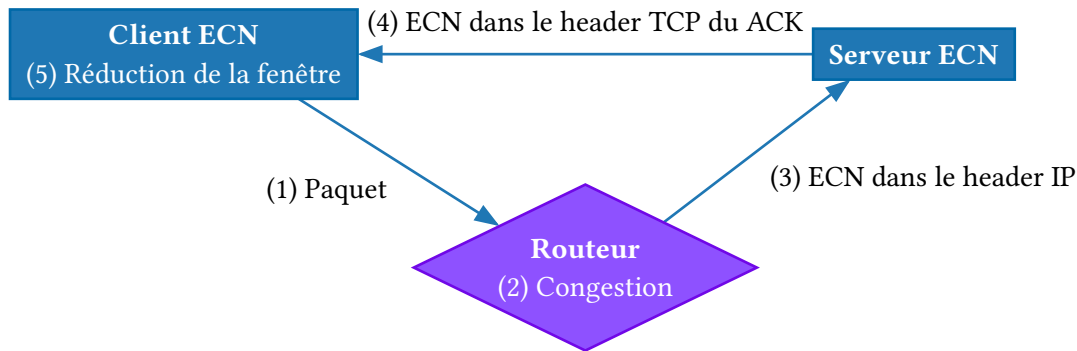


Fig. 5. – Fonctionnement d'ECN sur un aller retour de paquets

L4S (Low Latency, Low Loss and Scalable Throughput)

Architecture Internet rendant l'usage d'ECN compatible avec l'Internet public au moyen d'une double queue qui sépare le trafic classique du trafic L4S

L4S tente donc d'atteindre même en cas de congestion un délai faible sans perte de paquets (via l'utilisation d'ECN), qui s'adapte à la mise à l'échelle (indépendance vis à vis de la taille de la queue).

L4S utilise la double queue DualPI2 afin de séparer et réguler les 2 types de trafic traités.

Délai de queuing moyen $< 1\text{ms}$ et 99e pourcentile $< 2\text{ ms}$ [6]

DualPI2 AQM

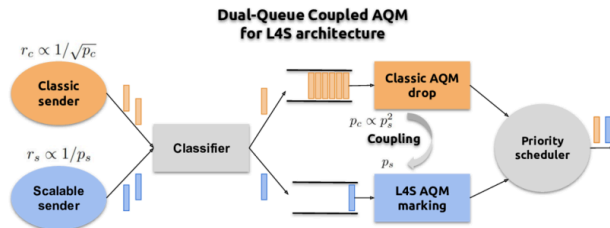


Fig. 6. – Schéma de principe de l'AQM DualPI2

La double queue permet de traiter le trafic L4S indépendamment du trafic « classique ».

Pour garantir l'équité entre ces 2 queues, un système de couplage est mis en place qui permet de marquer/drop des paquets dans chaque file selon l'état de l'autre, ainsi qu'un *scheduler* qui permet d'envoyer en alternance des paquets de chaque file.

TCP Prague

Algorithme de congestion utilisé par L4S.

Les exigences Prague :

1. Utilisation d'ECN

- Feedback AccECN
- Utilisation spécifique du bit de marquage ECN(1)
- Utilisation d'une fenêtre de congestion fractionnaire

2. Comportement Reno-friendly

- Compatibilité Reno en cas de perte
- Compatibilité Reno face à une congestion ECN

3. Nuances supplémentaires dans le traitement des données

- Réduction de la dépendance à la latence
- Détection de la perte en unité de temps

Il existe aussi d'autres protocoles appliquant ces exigences hors TCP, comme UDP Prague.

On peut donc bénéficier des faibles délais de L4S sur tout type de connexion.

AccECN : (RFC 9768, Avril 2026 [7]) une extension d'ECN qui permet de réutiliser des champs obsolètes du header TCP (CWR et NS) pour donner d'avantage de précisions sur le degré de congestion

3 État de l'art

Les RFCs fondatrices

Trio de RFC publié par l'IETF en 2023, co-écrit par Bob Briscoe, Koen De Schepper, Marcelo Bagnulo et Greg White :

RFC 9330 *L4S Internet Service: Architecture* [6]

RFC 9331 *The ECN Protocol for L4S* [8]

RFC 9332 *Dual-Queue Coupled AQM for L4S* [9]

Fondations : **CoDel** (contrôle par temps de séjour, RFC 8289 [10]), et **FQ-CoDel** (file par flux, RFC 8290 [11]).

Implémentations commerciales

Apple est le premier éditeur grand public à avoir déployé L4S.

- **iOS 16+ / macOS Ventura** : support natif activé par défaut
- **WWDC23** « *Reduce network delays with L4S* » : démonstration des gains en latence pour les appels vidéo, jeux et streaming avec un PoC
- **Network framework** : API pour tester L4S dans ses propres apps (`NWParameters.isL4SEnabled`)
- Outil de debug dédié : profil de configuration réseau pour simuler un réseau L4S en lab

Implémentations commerciales — cont.

Android : intégration en cours dans AOSP

- Travaux visibles dans les patches noyaux Android pour supporter DualPI2 et AccECN
- Cible : réduire la latence des applications temps-réel (gaming, visio)

NVIDIA : L4S dans GeForce NOW :

- Complète l'approche NVIDIA Reflex (réduction de la latence côté PC)

Comcast (NANOG 2024) :

- Déploiement pilote L4S sur infrastructure FAI réelle
- Résultats terrain : réduction drastique du bufferbloat pour les abonnés

Kernel Linux

La L4S Team maintient les patchs L4S pour le kernel

- **TCP Prague** (CCA aux endpoints)
- **DualPI2** (gestion de la queue d'un routeur)
- Sur Github : [L4STeam/linux](https://github.com/L4STeam/linux) [12], [13]

L'acceptation de ces patchs sur le tronc Linux sera une étape majeure de l'adoption de L4S par l'industrie !

L4S Team maintient aussi :

- patchs pour `iproute2` [14]
- **UDP-Prague** : support L4S pour des applications UDP (QUIC, WebRTC, etc.) [15]

Travaux de recherche

Briscoe et al. [16] : expérimentations sur TCP Prague et L4S :

- Latence divisée par 10 comparé aux flux Reno-friendly sous charge
- Zéro perte supplémentaire pour les flux classiques en coexistence

Olga Albisser et al. [17], [18] : preuves de concept sur environnements émulés :

- Validation du comportement de DualPI2 avec des flux Prague/CUBIC en parallèle
- Mesure de l'équité inter-flux et de l'impact sur la latence

Évaluation de cas d'usage spécifiques :

- Visioconférence volumétrique bas-débit en réseau congestionné (L4S réduit les artefacts) [19]
- Transmission vidéo temps-réel dans des réseaux **5G industriels privés** (cf. DCTCP) [20]
- Évaluation sur **dataplane programmable** (P4/DPDK) [21]

Outil de mesure existant : aqmt [22]

- Framework conteneurisé pour tester des AQMs
- **Limites**, pour l'architecture L4S : aqmt est couplé à une **topologie spécifique** (deux clients et deux serveurs) et **ne supporte pas nativement la dualpi2 AQM**

Objectifs

Créer un **environnement de test reproductible** permettant de **mesurer et comparer** le **comportement de l'algorithme de contrôle de congestion TCP Prague** en coexistence avec Cubic et BBR, dans l'**architecture L4S**

En détails :

- Mettre en place un environnement
 - **Reproductible et paramétrable** (bande passante, délai...)
 - Étudiant un routeur **DualPI2 AQM**
 - Avec des clients TCP Prague, Cubic et BBR
- Obtenir des mesures en **haute fréquence** de la latence, de la gigue, du débit et des pertes
- Évaluer l'**équité** inter-algorithmes et la coexistence Prague / flux classiques

4 Nos contributions

Le framework PRIM-L4S

Objectifs :

- Créer un framework permettant l'**étude du comportement de TCP Prague, Cubic & BBR** au contact d'un **routeur dualpi2 AQM**
- Qui devrait être **scalable** en :
 - Nombre de clients
 - Quantité de données récoltées
 - Variété de scénarios

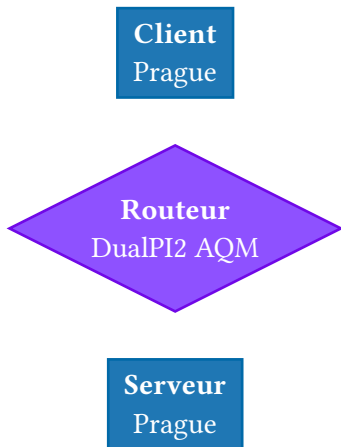
Le framework PRIM-L4S est disponible

- Sur Github : [PRIM-L4S/PRIM-L4S](#) [23]
- Sous licence **MIT** (très permissive)

Les cinq composants de notre framework

- **Banc de tests conteneurisés** (*Docker* 🐳) : déploiement d'une infrastructure avec clients, serveurs et routeur
- **High Frequency Exporter** (*Rust* 🦀) : collection de métriques TCP à **haute fréquence**
- **Scenarii generator** (*Rust* 🦀) : planification de la matrice d'expériences à exécuter
- **Serial Experiment Runner** (*Rust* 🦀) : orchestration de l'exécution des **200 expériences**
- **Result analysis** (*Python* 🐍) : analyse des données et production des courbes

Banc de tests conteneurisé : Client, serveur et routeur

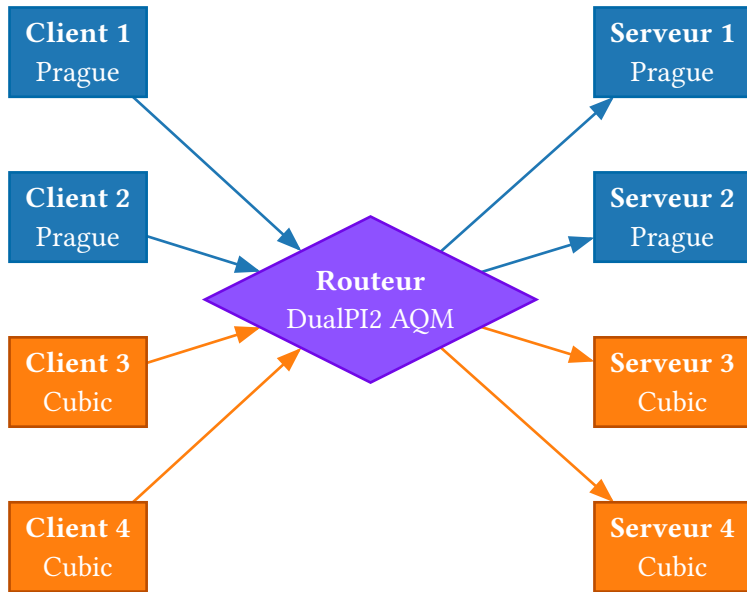


- **Objectif** : créer un banc de test **scalable** et **flexible** où étudier l'architecture L4S
- Dans notre infrastructure, trois types d'éléments : client, serveur, et routeur
- Tous trois sont **conteneurisés** avec Docker
- Installation du **patch kernel L4S** sur la machine hôte
- Les clients et serveurs supportent : TCP **Prague**, **Cubic** et **BBR**

Trois images Docker créées

- **Client** : un client iperf3 supportant L4S et l'exécutable High Frequency Exporter
- **Serveur** : un serveur iperf3 supportant L4S
- **Routeur** : un routeur Linux avec la queue L4S **Dualpi2 AQM** et l'exécutable High Frequency Exporter

Banc de tests conteneurisé : Topologie



- Benchmark **Iperf3** du client *i* vers le serveur *i*
- Déployable sur **une** machine ou **plusieurs** (scalabilité). Pour nous, une seule machine

Les scénarios

- **Scénario** : un fichier (JSON) qui décrit la topologie de l'expérience à mener : le nombre de clients/serveurs, leur l'algorithme de congestion, etc...
- Le banc de tests conteneurisé prend **en entrée un scénario et déploie l'infrastructure de test correspondante**

High Frequency Exporter : Obtention des métriques

Client 1 Prague

- **Objectif** : obtenir les métriques réseau avec une « grande fréquence »

Client 2 Prague

- Métriques observées :
 - Latence
 - Gigue
 - Débit
 - Taille des fenêtres de congestion
 - Taille des queues du routeur
 - Et **18 autres...**

Client 3 Cubic

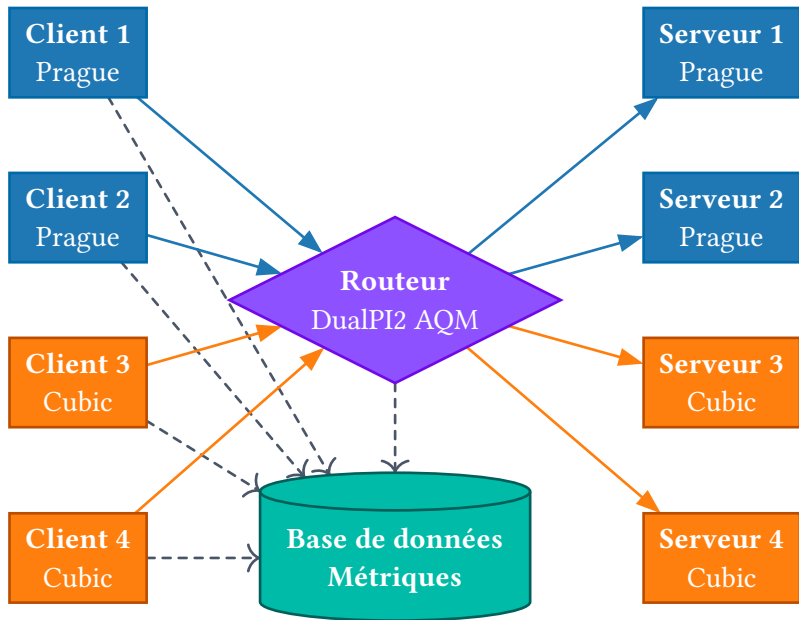
- Précision : **1 point toutes les millisecondes**

Client 4 Cubic

Appels direct au kernel Linux (sur les clients)

- Initialisation
 - Obtenir l'*inode* associé au socket du benchmark iperf3
 - Obtenir le *process ID* de iperf3 et le *file descriptor* associé au socket
 - Cloner le *file descriptor* dans le processus du High Frequency Exporter
- Boucle haute fréquence (1 point / ms)
 - Appeler `libc::getsockopt` sur le *file descriptor* cloné
 - On obtient les métriques du kernel sur le socket (TCP_INFO)

High Frequency Exporter : Stockage des métriques

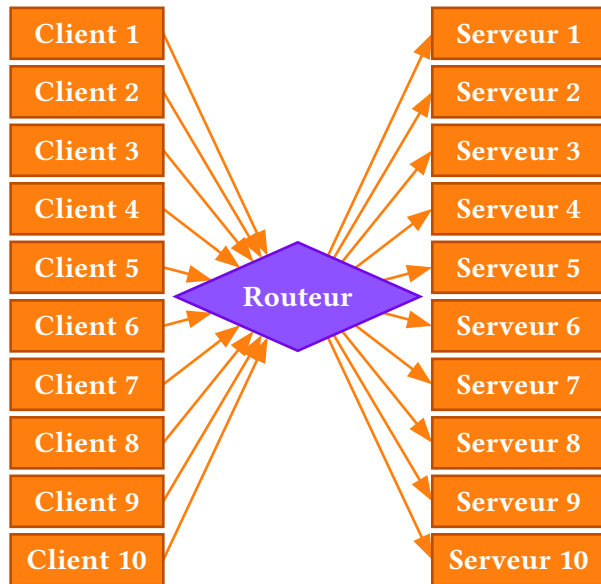


- Comment exporter et stocker un **grand volume** de métriques de manière **scalable** et **flexible** ?
- Deux étapes :
 - ▶ **Emmagasiner** (1 fois / ms) : sur chaque client, stocker temporairement les métriques
 - ▶ **Exporter** (1 fois / 10 s) : chaque client envoie les données des **10 dernières secondes** au conteneur de base de données central
- Conteneur de base de données : **Victoria Metrics** (dérivé de Prometheus, un format standard)

Nos expériences

Paramètres

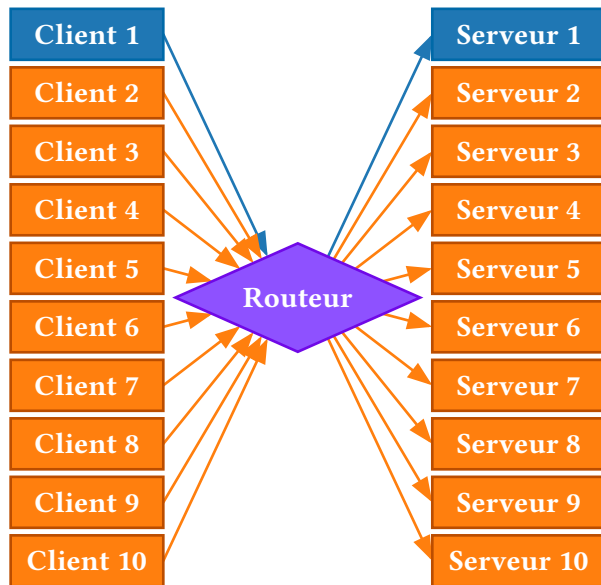
- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - Prague vs. Cubic (exemple à droite)
 - Prague vs. BBR
 - Cubic vs. BBR
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : **200**
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**



Nos expériences

Paramètres

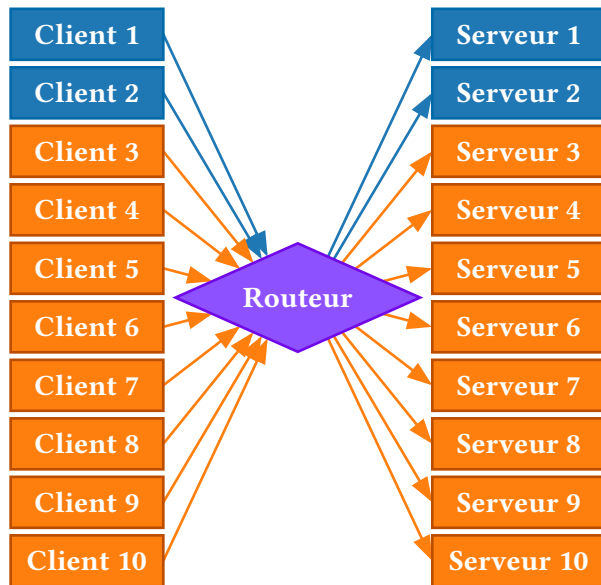
- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - Prague vs. Cubic (exemple à droite)
 - Prague vs. BBR
 - Cubic vs. BBR
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : 200
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**



Nos expériences

Paramètres

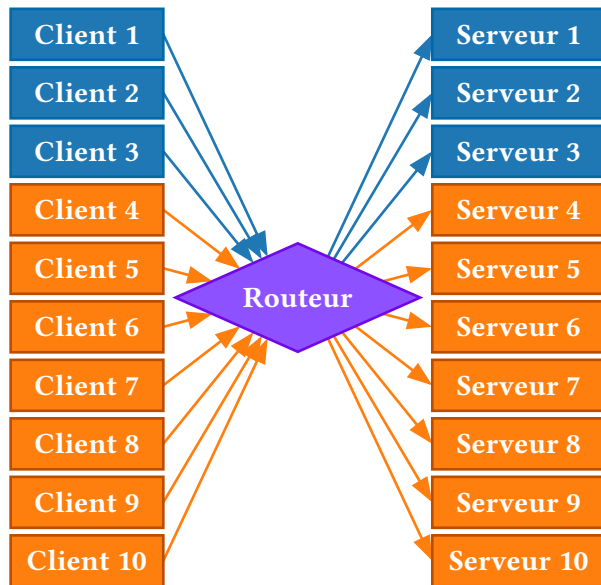
- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - Prague vs. Cubic (exemple à droite)
 - Prague vs. BBR
 - Cubic vs. BBR
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : 200
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**



Nos expériences

Paramètres

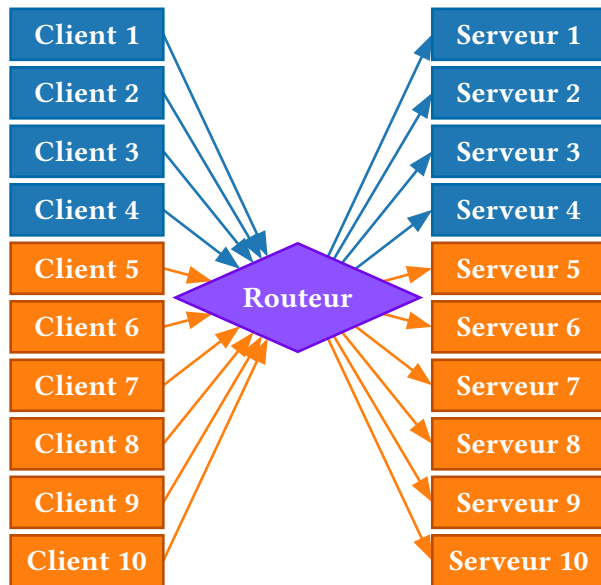
- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - Prague vs. Cubic (exemple à droite)
 - Prague vs. BBR
 - Cubic vs. BBR
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : **200**
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**



Nos expériences

Paramètres

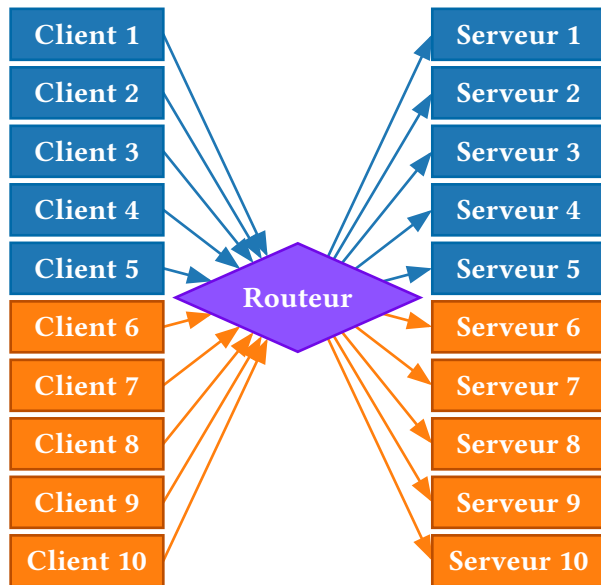
- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - Prague vs. Cubic (exemple à droite)
 - Prague vs. BBR
 - Cubic vs. BBR
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : 200
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**



Nos expériences

Paramètres

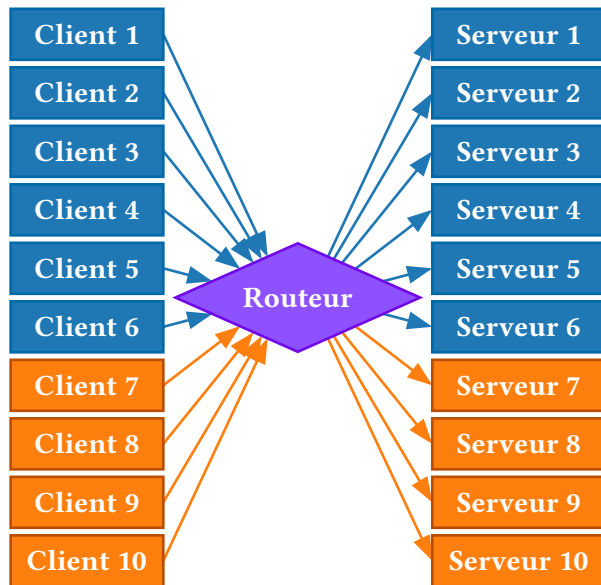
- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - Prague vs. Cubic (exemple à droite)
 - Prague vs. BBR
 - Cubic vs. BBR
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : 200
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**



Nos expériences

Paramètres

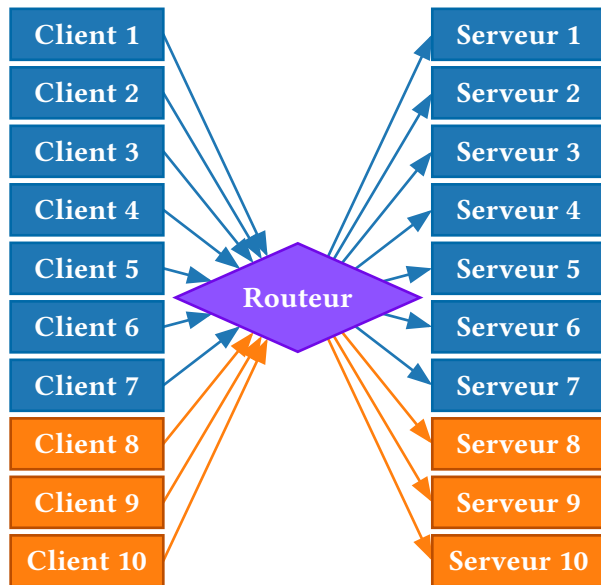
- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - Prague vs. Cubic (exemple à droite)
 - Prague vs. BBR
 - Cubic vs. BBR
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : 200
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**



Nos expériences

Paramètres

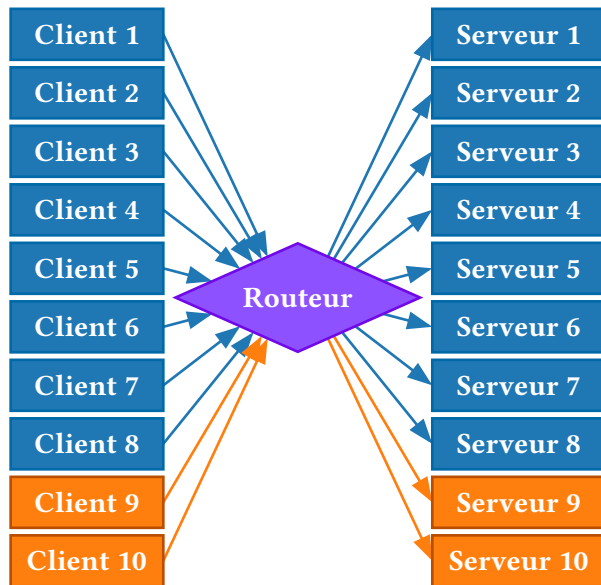
- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - **Prague** vs. **Cubic** (exemple à droite)
 - **Prague** vs. **BBR**
 - **Cubic** vs. **BBR**
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : **200**
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**



Nos expériences

Paramètres

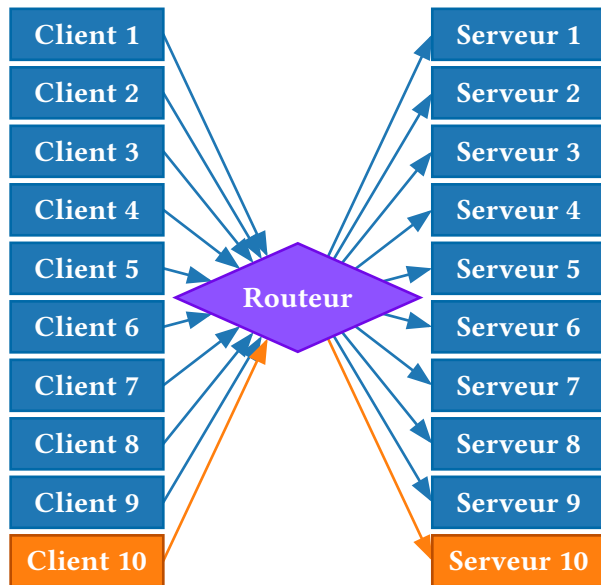
- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - Prague vs. Cubic (exemple à droite)
 - Prague vs. BBR
 - Cubic vs. BBR
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : 200
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**



Nos expériences

Paramètres

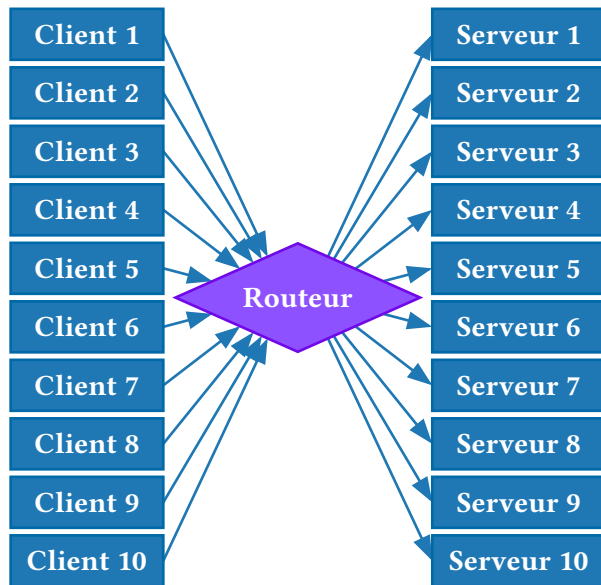
- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - Prague vs. Cubic (exemple à droite)
 - Prague vs. BBR
 - Cubic vs. BBR
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : 200
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**



Nos expériences

Paramètres

- Toujours 10 clients, 10 serveurs et 1 routeur
 - Comparaisons deux à deux:
 - Prague vs. Cubic (exemple à droite)
 - Prague vs. BBR
 - Cubic vs. BBR
 - Une **limite de débit** imposée au routeur à 1 Gbit/s, 100 Mbit/s, 10 Mbit/s ou 5 Mbit/s
 - Un **délai ajouté** en entrée du routeur de 10 ms
 - Chaque tour : 6×40 s de benchmarks
-
- Nombre d'expériences réalisées : **200**
 - Planificateur: **Scenarii generator**
 - Orchestrateur : **Serial Experiment Runner**





Démonstration

Présentation du framework PRIM-L4S en fonctionnement

5 Nos résultats

Comprendre nos courbes

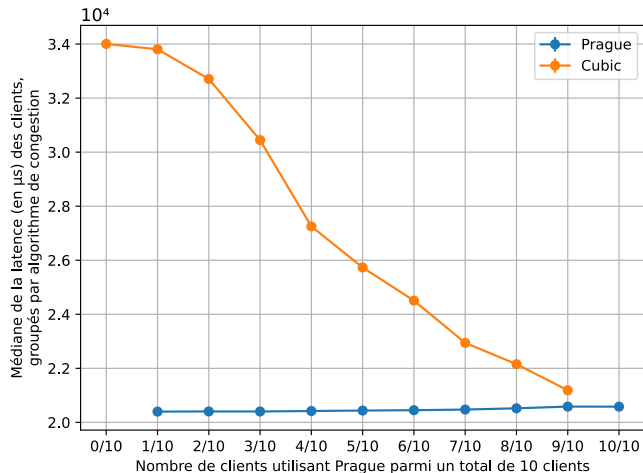
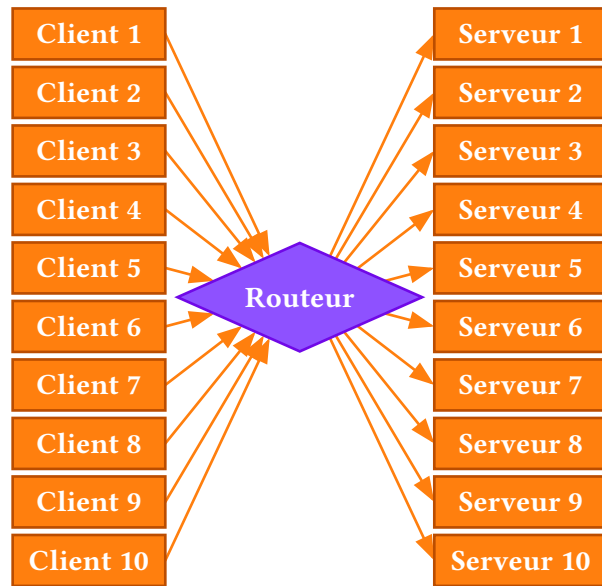


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Comprendre nos courbes

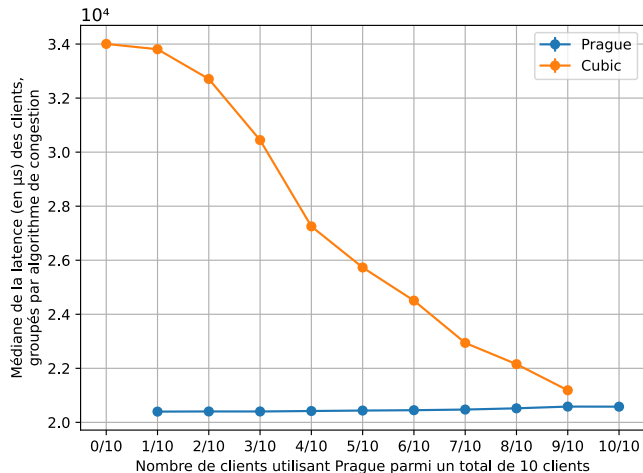
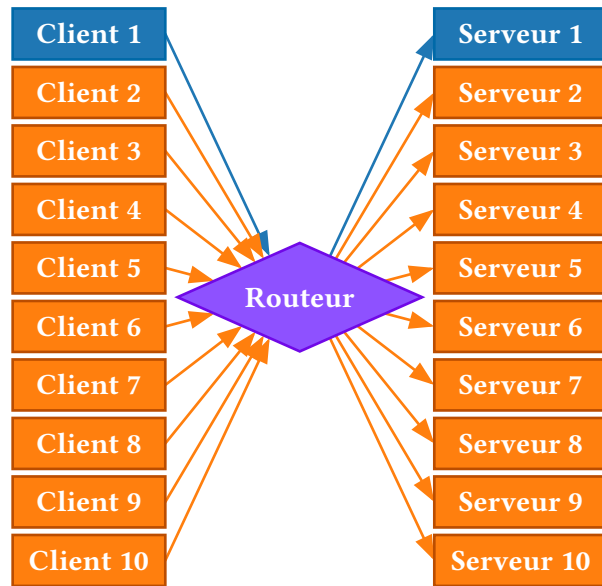


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Comprendre nos courbes

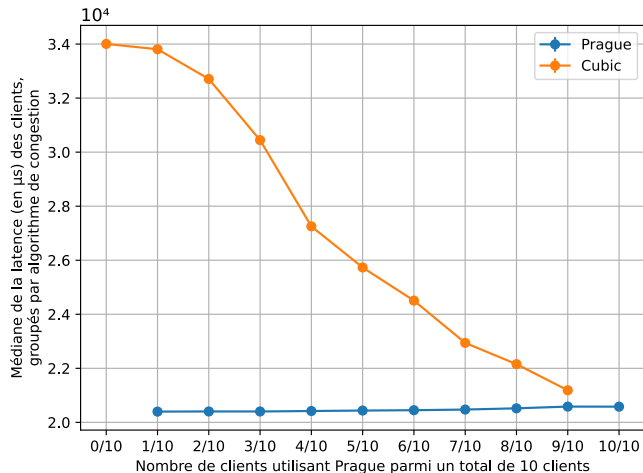
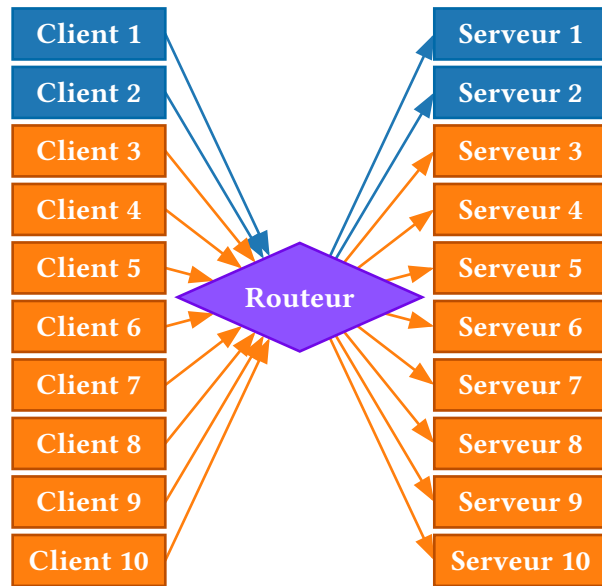


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Comprendre nos courbes

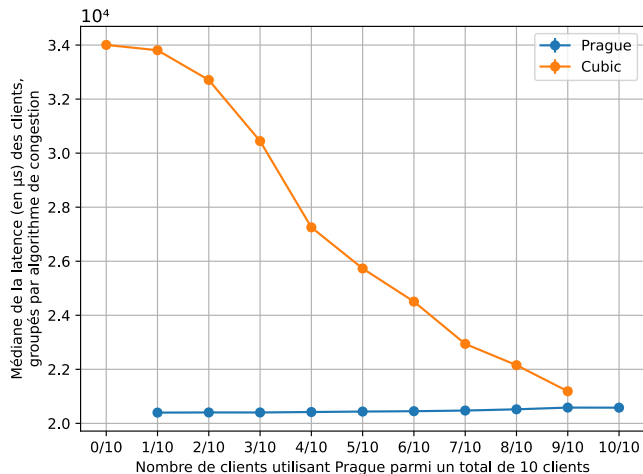
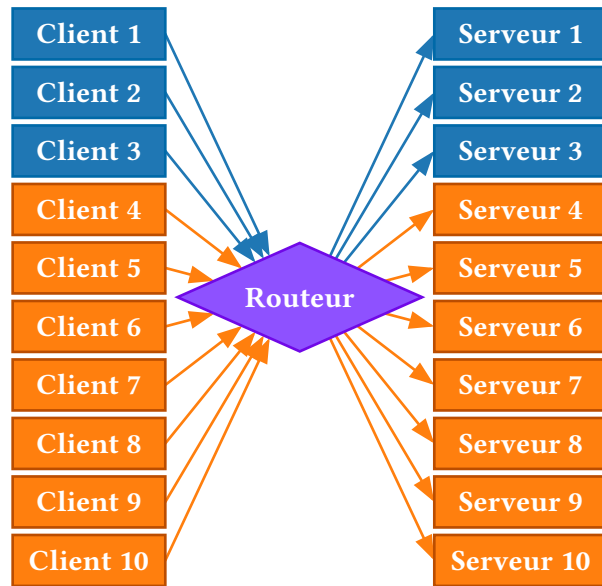


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Comprendre nos courbes

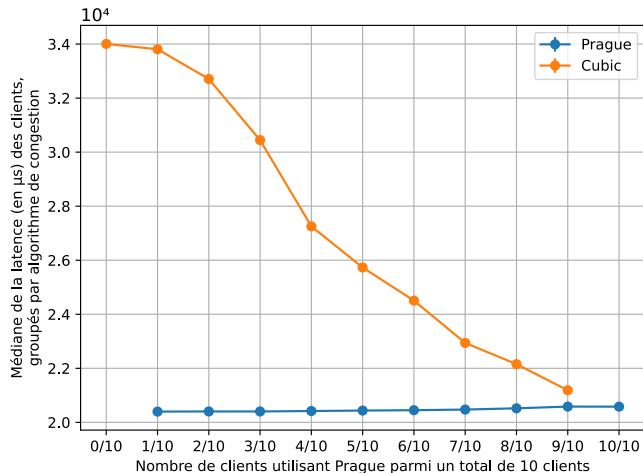
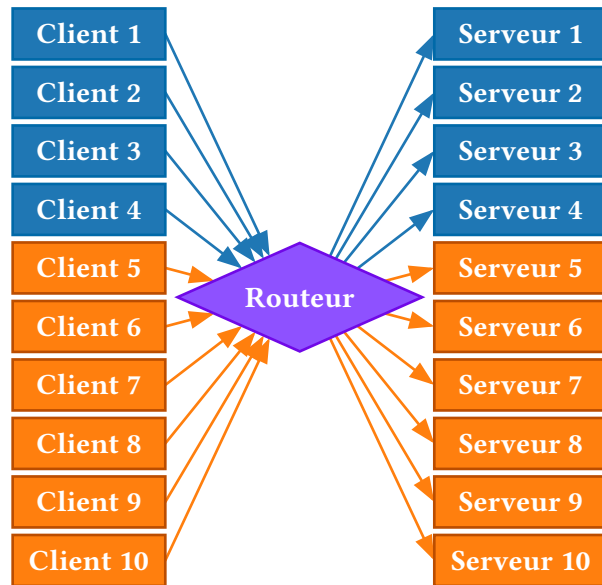


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Comprendre nos courbes

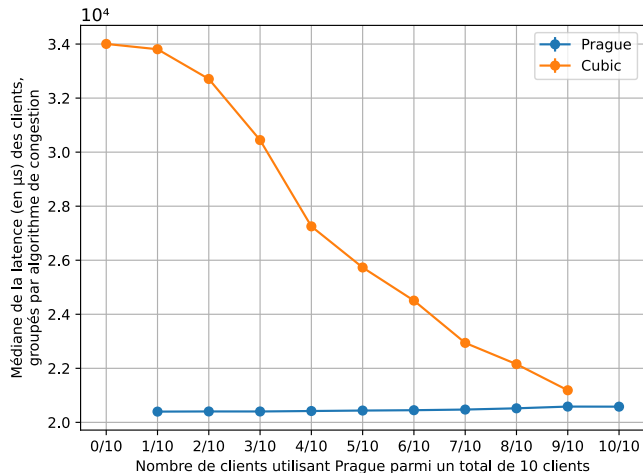
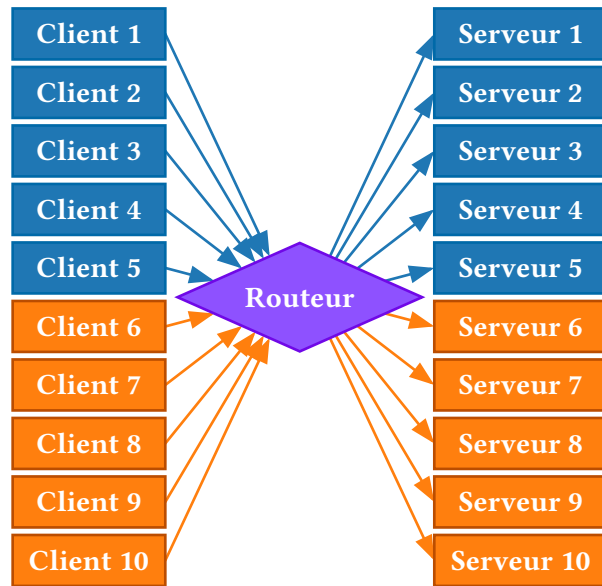


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Comprendre nos courbes

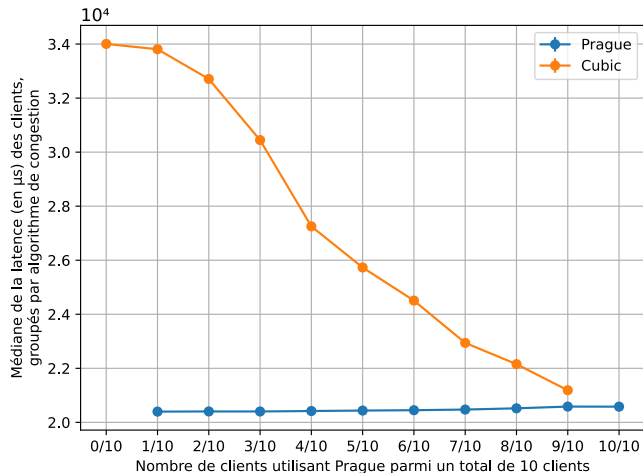
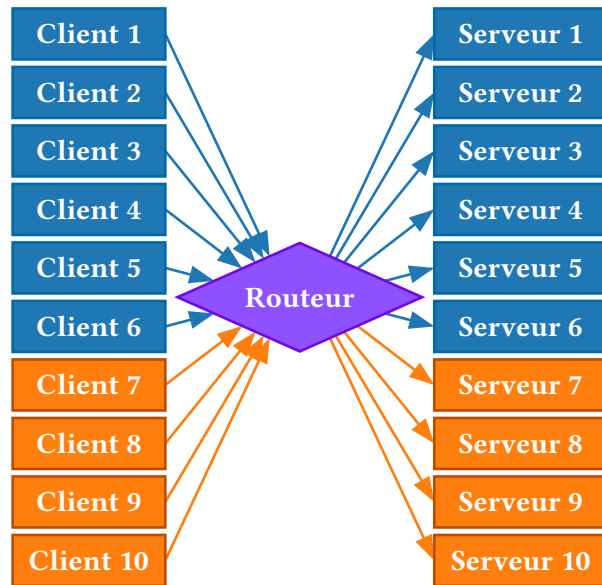


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Comprendre nos courbes

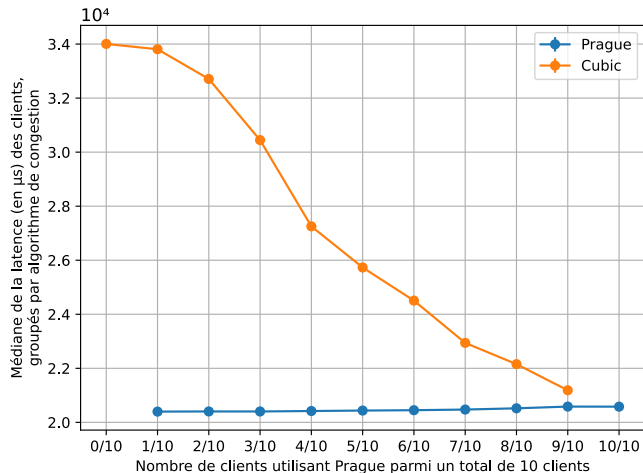
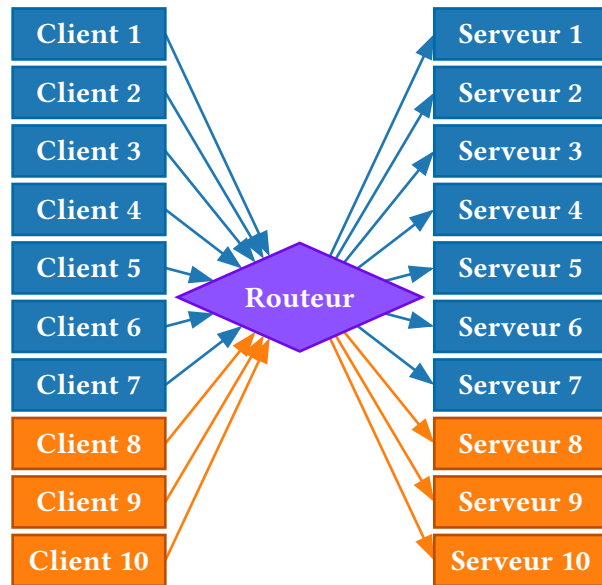


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Comprendre nos courbes

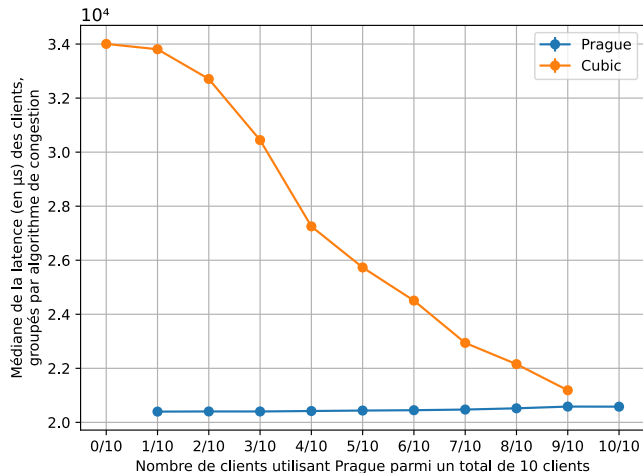
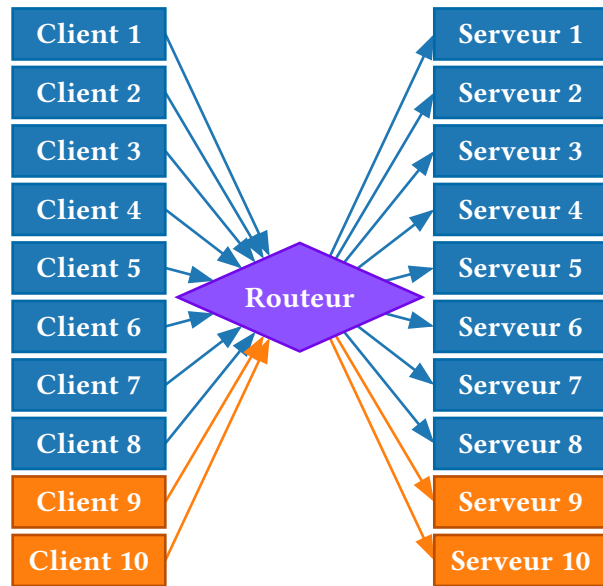


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Comprendre nos courbes

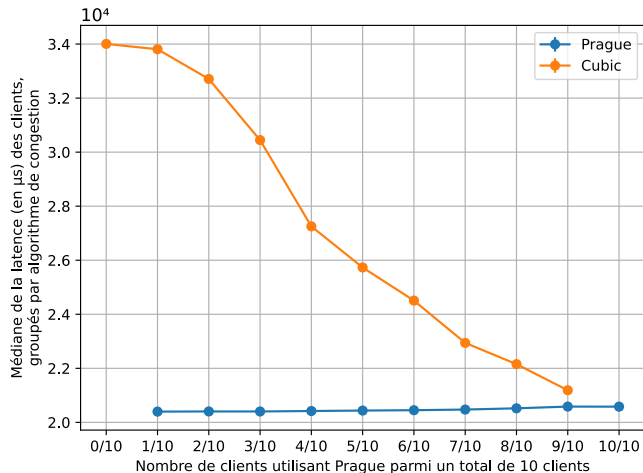
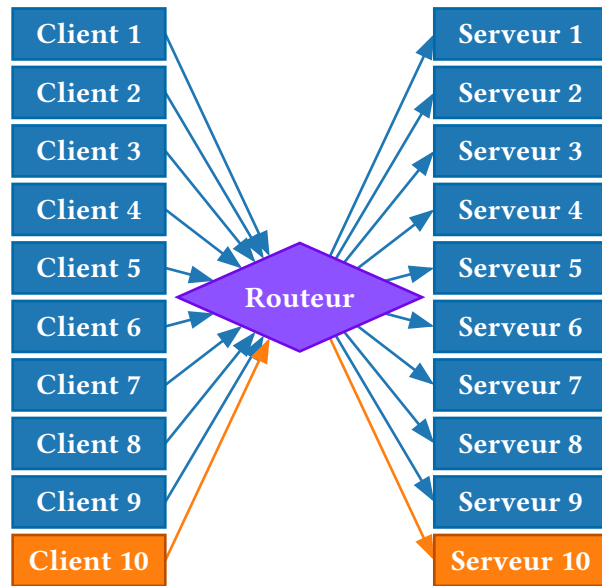


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Comprendre nos courbes

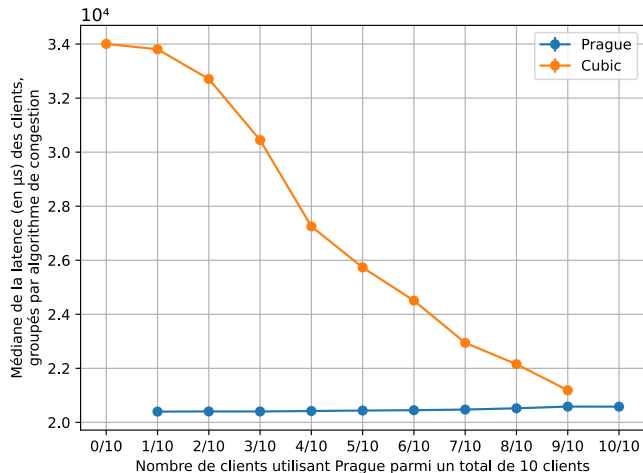
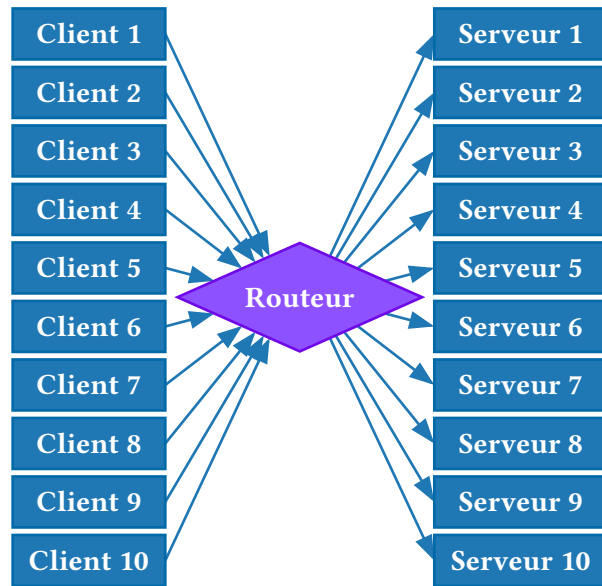


Fig. 7. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**



Résultats (Latence)

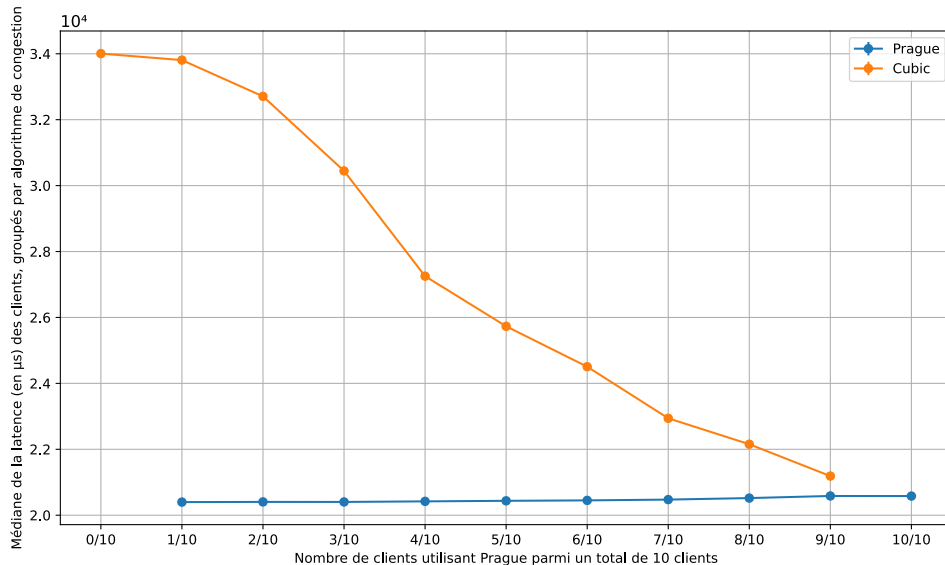


Fig. 8. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **Cubic**

Résultats (Latence)

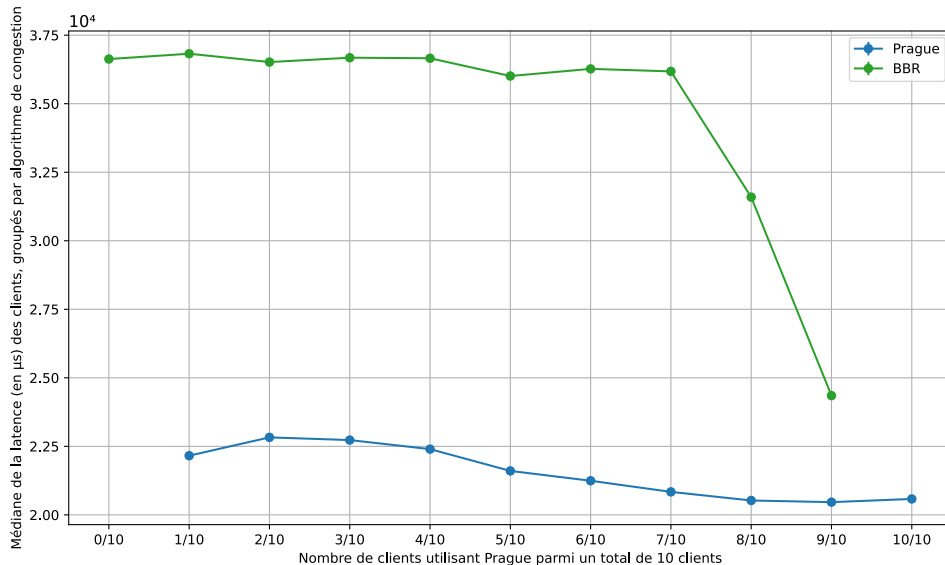


Fig. 9. – Comparaison de la **latence** médiane des clients utilisant **Prague** et **BBR**

Résultats (Gigue)

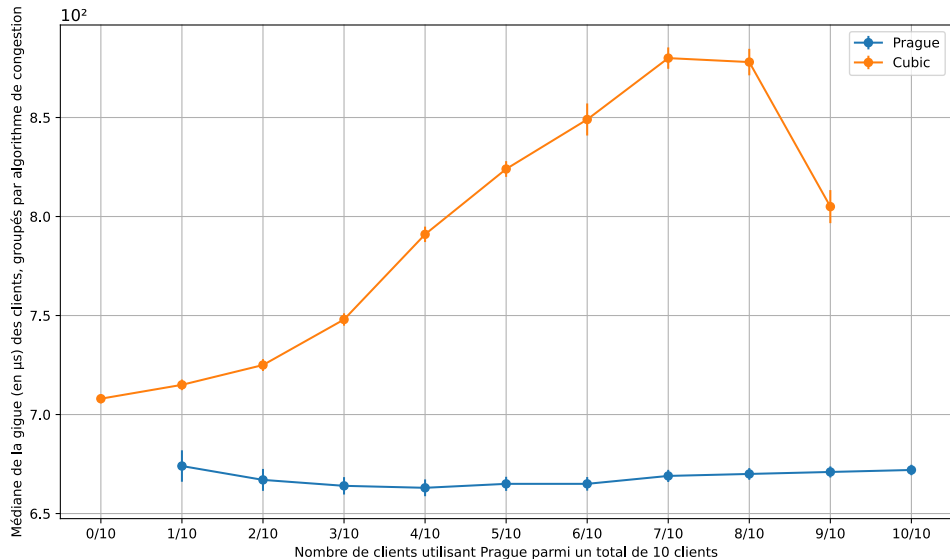


Fig. 10. – Comparaison de la **gigue** médiane des clients utilisant **Prague** et **Cubic**

Résultats (Gigue)

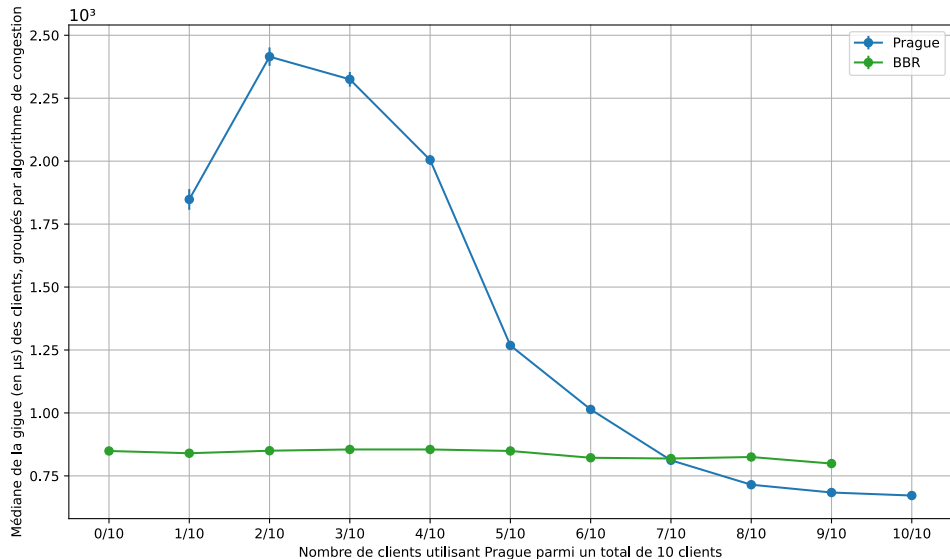


Fig. 11. – Comparaison de la **gigue** médiane des clients utilisant **Prague** et **BBR**

Instabilité de Prague ou agressivité de BBR ?

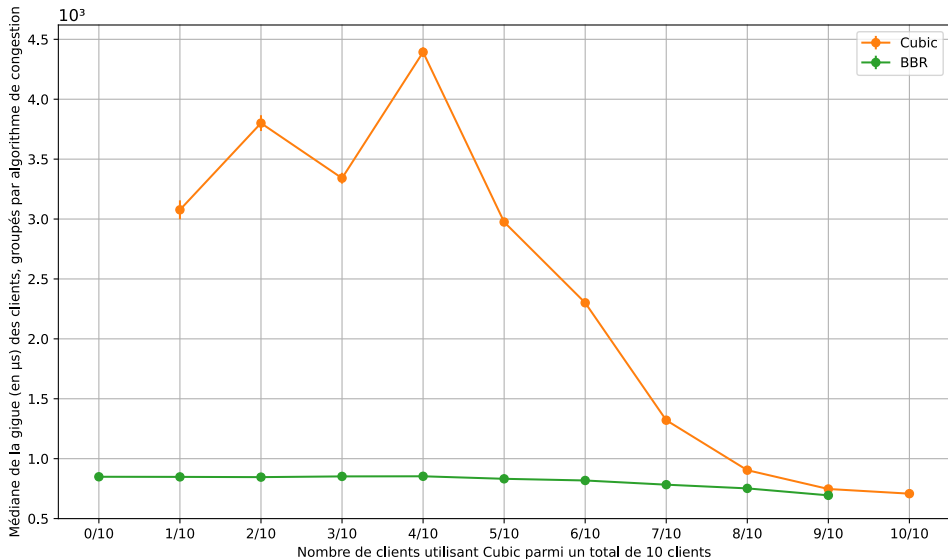


Fig. 12. – Comparaison de la **gigue** médiane des clients utilisant **Cubic** et **BBR**

Résultats (Débit)

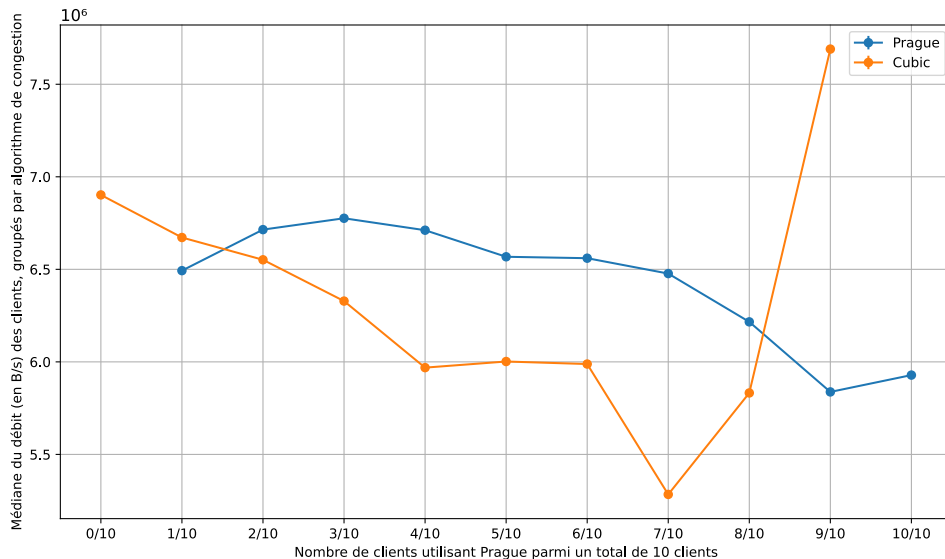


Fig. 13. – Comparaison du **débit** médian des clients utilisant **Prague** et **Cubic**

Résultats (Débit)

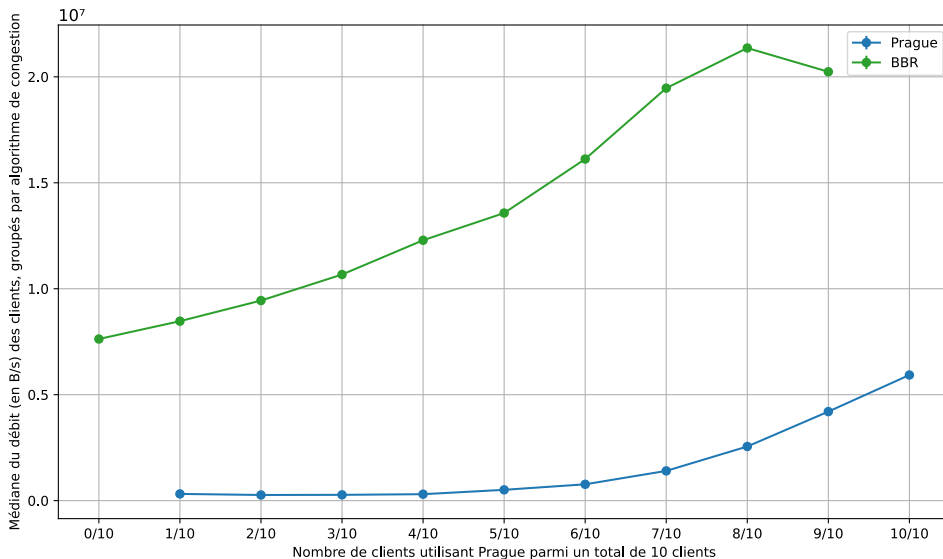


Fig. 14. – Comparaison du **débit** médian des clients utilisant **Prague** et **BBR**

Faible débit de Prague ou agressivité de BBR ?

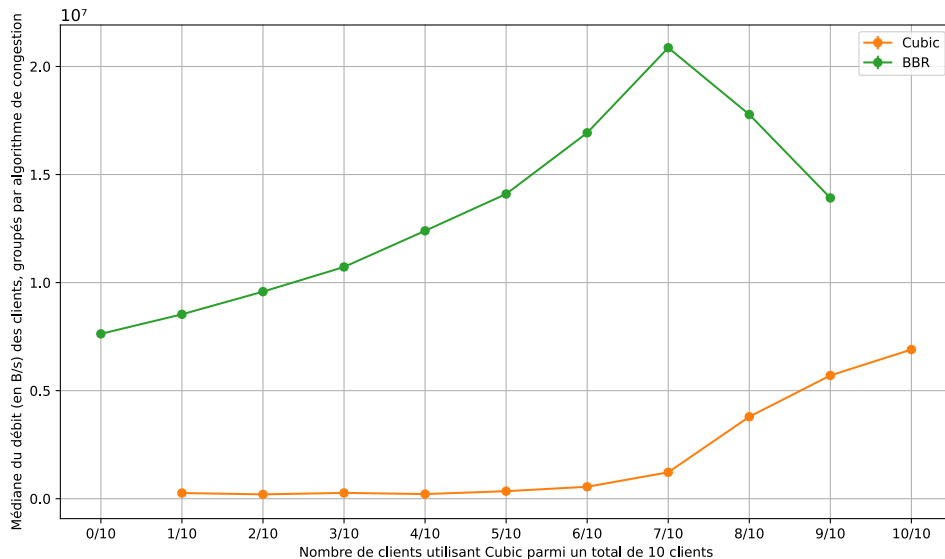


Fig. 15. – Comparaison du **débit** médian des clients utilisant **Cubic** et **BBR**

En résumé

Dans **cette topologie** avec un routeur **dualpi2 AQM**, dans un réseau **conteneurisé**, on observe que :

Algorithme comparé à Prague	Latence	Gigue	Débit
			=
Cubic	✓	✓	Note : Parfois meilleur ou moins bon en fonction du nombre de clients
		✗	✗
BBR	✓	Note : Prague et Cubic partagent ce défaut, en comparaison à BBR	Note : Prague et Cubic partagent ce défaut, en comparaison à BBR

Signification : la latence de **Prague** est meilleure (✓) que celle de **Cubic**

6 Conclusion

Conclusion

Nos contributions : le framework PRIM-L4S

- Permet l'étude du comportement de TCP **Prague**, Cubic & BBR dans l'architecture **L4S**
 - Un framework **scalable** en :
 - Nombre de clients simulés
 - Quantité de données récoltée
 - Variété de scénarios étudié
 - Qui nous a permis de **mieux comprendre le comportement de TCP Prague** lorsque la proportion de clients ayant fait la transition évolue
-
- Un grand merci à Jean-Louis, Nadia, Davor   et Sébastien !

Le framework PRIM-L4S est disponible

- Sur Github : [PRIM-L4S/PRIM-L4S](#) [23]
- Sous licence **MIT** (très permissive)

N'hésitez pas à nous contacter en cas de besoin !

- **Emre Ucar**
emre@melix.net
- **Thomas Sauvage**
thomas@sauvage.pm
- **Timothée Fisher**
timothee.fisher.2022@polytechnique.org

Références

- [1] A. Gurtov, T. Henderson, S. Floyd, et Y. Nishida, « The NewReno Modification to TCP's Fast Recovery Algorithm », Request for Comments RFC6582, avr. 2012. Consulté le: 29 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/rfc6582/>
- [2] L. Xu, S. Ha, I. Rhee, V. Goel, et L. Eggert, « CUBIC for Fast and Long-Distance Networks », Request for Comments RFC9438, août 2023. Consulté le: 29 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/rfc9438/>
- [3] N. Cardwell, I. Swett, et J. Beshay, « BBR Congestion Control », Internet Draft draft-ietf-ccwg-bbr-05, mars 2026. Consulté le: 29 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/draft-ietf-ccwg-bbr/>
- [4] S. Floyd, K. K. Ramakrishnan, et D. L. Black, « The Addition of Explicit Congestion Notification (ECN) to IP », Request for Comments RFC3168, sept. 2001. Consulté le: 29 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/rfc3168/>

Références (ii)

- [5] S. Bensley, D. Thaler, P. Balasubramanian, L. Eggert, et G. Judd, « Data Center TCP (DCTCP): TCP Congestion Control for Data Centers », Request for Comments RFC8257, oct. 2017. Consulté le: 29 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/rfc8257/>
- [6] B. Briscoe, K. D. Schepper, M. Bagnulo, et G. White, « Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture », Request for Comments RFC9330, janv. 2023. Consulté le: 23 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/rfc9330/>
- [7] B. Briscoe, M. Kühlewind, et R. Scheffenegger, « More Accurate Explicit Congestion Notification (AccECN) Feedback in TCP », Request for Comments RFC9768, avr. 2026. Consulté le: 29 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/rfc9768/>
- [8] K. D. Schepper et B. Briscoe, « The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S) », Request for Comments RFC9331, janv. 2023. Consulté le: 23 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/rfc9331/>

Références (iii)

- [9] K. D. Schepper, B. Briscoe, et G. White, « Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S) », Request for Comments RFC9332, janv. 2023. Consulté le: 23 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/rfc9332/>
- [10] K. Nichols, V. Jacobson, A. McGregor, et J. Iyengar, « Controlled Delay Active Queue Management », Request for Comments RFC8289, janv. 2018. Consulté le: 23 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/rfc8289/>
- [11] T. Høiland-Jørgensen, P. McKenney, dave.taht@gmail.com, J. Gettys, et E. Dumazet, « The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm », Request for Comments RFC8290, janv. 2018. Consulté le: 23 juin 2026. [En ligne]. Disponible sur: <https://datatracker.ietf.org/doc/rfc8290/>
- [12] « L4STeam/linux ». Consulté le: 23 juin 2026. [En ligne]. Disponible sur: <https://github.com/L4STeam/linux>
- [13] « L4STeam/linux-net-next ». Consulté le: 23 juin 2026. [En ligne]. Disponible sur: <https://github.com/L4STeam/linux-net-next>

Références (iv)

- [14] « L4STeam/iproute2 ». Consulté le: 23 juin 2026. [En ligne]. Disponible sur: <https://github.com/L4STeam/iproute2>
- [15] « L4STeam/udp_prague ». Consulté le: 23 juin 2026. [En ligne]. Disponible sur: https://github.com/L4STeam/udp_prague
- [16] B. Briscoe *et al.*, « Implementing the 'Prague Requirements' for Low Latency Low Loss Scalable Throughput (L4S) », présenté à Netdev 0x13, 2019. [En ligne]. Disponible sur: <https://bobbriscoe.net/projects/latency/tcp-prague-netdev0x13.pdf>
- [17] O. Albisser, K. De Schepper, B. Briscoe, O. Tilmans, et H. Steen, « DUALPI2 - Low Latency, Low Loss and Scalable Throughput (L4S) AQM », présenté à Netdev 0x13, 2019. [En ligne]. Disponible sur: https://www.bobbriscoe.net/projects/latency/dualpi2_netdev0x13.pdf
- [18] « Netdev 0x13 - DUALPI2 Low Latency, Low Loss and Scalable L4S AQM ». Consulté le: 30 juin 2026. [En ligne]. Disponible sur: <https://www.youtube.com/watch?v=g9VfnNTDv-I>

Références (v)

- M. De Fré *et al.*, « Low-Latency Volumetric Video Conferencing in Congested Networks Through L4S », in *Proceedings of the 16th ACM Multimedia Systems Conference*, Stellenbosch South Africa: ACM, mars 2025, p. 113-123. doi: [10.1145/3712676.3714443](https://doi.org/10.1145/3712676.3714443).
- [19]
- L. V. Monteiro, V. S. Simão, R. d. B. Lira, L. C. de Almeida, R. D. Gomes, et P. Ditarso Maciel, « L4S in Private 5G Industrial Networks: A Case Study for Real-Time Video Transmission in Programmable Networks », in *2024 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, nov. 2024, p. 1-4. doi: [10.1109/NFV-SDN61811.2024.10807467](https://doi.org/10.1109/NFV-SDN61811.2024.10807467).
- [20]
- L. C. D. Almeida, P. D. Maciel Jr., et F. L. Verdi, « Evaluating L4S Framework Performance with Programmable Data Plane Hardware », in *Anais do XXIX Workshop de Gerência e Operação de Redes e Serviços (WGRS 2024)*, Brasil: Sociedade Brasileira de Computação - SBC, mai 2024, p. 196-209. doi: [10.5753/wgrs.2024.3286](https://doi.org/10.5753/wgrs.2024.3286).
- [21]
- [22] H. Steen, « [henrist/aqmt](https://github.com/henrist/aqmt) ». Consulté le: 23 juin 2026. [En ligne]. Disponible sur: <https://github.com/henrist/aqmt>

Références (vi)

- [23] « PRIM-L4S/PRIM-L4S ». Consulté le: 23 juin 2026. [En ligne]. Disponible sur: <https://github.com/PRIM-L4S/PRIM-L4S>